GRAPHICAL RESOURCE EDITOR FOR SOFTWARE CUSTOMIZATION

This application is a continuation of application Ser. No. 07/941,579, filed Sep. 8, 1992, now abandoned.

BACKGROUND OF THE INVENTION

The present invention relates generally to the customization of graphically-controlled software in a data processing system. More particularly, the invention pertains to an editing system for selectively modifying resources used by software applications to provide graphical user interface objects.

A graphical user interface is a window-based system for interfacing with a programmed computer by means of graphical display windows. Such systems are ideally suited and conventionally adapted for menu-driven operation. That is, they allow a user to control computer operation by selecting from one or more menus exhibited in the graphical display windows without entering statements in alphanumeric form via a keyboard. A graphical display window is an area of the computer's physical display screen containing a collection of interface objects through which software applications running on the computer receive inputs and present their output. An interface object, such as a menu selection item, is a graphics- and/or text-based image that signifies information, function, and/or data entry. Such objects include push buttons, scroll bars, dials, sliders and many other graphical indicia. An interface object can be "selected" by moving a mouse-controlled pointer or cursor to it and operating (i.e., pressing or releasing) a button on the mouse. This selection process includes what is referred to in the art as "point-and-click". The interface object is supported by code that assists in providing appropriate response to the user's selection.

It is conventional for software applications to run in their own graphical display windows, the appearance and characteristics of which are defined by the application software. Each application employs one or more top level windows that define corresponding graphical display areas used by the application. In so-called "multitasking" operating systems, there may be several applications, each using one or more windows, sharing the screen together. To minimize confusion, these windows can be moved to different positions on the screen, and may also be resizable. Moreover, the windows are assigned a stacking order such that overlapping windows do not compromise window appearance. Higher order windows simply cover lower level windows in the area of overlap. Thus, the effect is that of sheets of paper arranged on a desktop.

One of the implications of a shared-screen, multitasking window environment is that the graphical display windows must provide a visually distinctive appearance that helps users control the sometimes complex functionality presented on the physical screen. Users may also have special requirements, apart from multitasking considerations, for ensuring particularized window characteristics. In the art of graphical user interfaces, the term "resources" refers to color, text fonts, screen cursors and other graphical attributes users may desire to control. Resource values are chosen by application developers during program development and may not represent an ideal choice when the application is implemented on a user's data processing system. Accordingly, it would be desirable to allow users to override or modify resource selections made by application programmers so that users

are able to customize efficiently the look and feel of their applications.

In some operating environments, user customization of window resources is intended. The X Window System is an industry-standard software system developed at Massachusetts Institute of Technology that provides a window environment for developing device-independent graphical user interfaces. In the X Window System, resources defining attributes of graphical interface objects are set by program developers in an application-specific file known as an "app-defaults" file. The X Window System allows users to override resource settings in the app-defaults file with a user-generated file known as an ".Xdefaults" file. When an application is invoked, the X Window System, in an initializing step, creates an resource database and loads it with resource specifications found in the application's app-default file and the user's .Xdefaults file. The resource database is used by the application to build its windows and graphical interface objects at run time.

In the X Window System, as in other graphical user interface systems, graphical interface objects are defined hierarchically. For applications based on the X toolkit intrinsics, each graphical interface object is constructed from one or more components known as "widgets". A widget is an object oriented programming entity containing data and one or more functions which operate on that data. When created by an application, a widget is assigned its own screen window. There are various standard widget sets available to application developers for building graphical user interfaces. Open Software Foundation, Inc. provides the Motif toolkit. Widgets in the Motif toolkit include scroll bars, title bars, menus, dialogue boxes and a host of other graphical interface objects.

In app-defaults and .Xdefaults files, the resources used by widgets are specified using hierarchical string identifiers that uniquely determine the widgets (or widget classes) to which they apply. These identifiers can include such information as the name (or class) of an application containing the widgets. Moreover, because widgets in an application's graphical display window are defined hierarchically, the resource identifier can include a list of widgets (or widget classes) arranged in the widget hierarchy. The syntax of these identifiers must be observed or the resource may not be correctly applied to the intended widgets (or widget classes). As a result, manipulating an .Xdefaults file can be cumbersome and time consuming.

In response to this dilemma, efforts have been made to develop editors for the X Window System that allow users to customize their applications. One such product is "Editres" from Massachusetts Institute of Technology. Editres is an editing tool that allows users to view the full widget hierarchy of any application that speaks the Editres protocol. Editres assists the user in constructing resource specifications and allows the user to apply the resource to an application and view the results dynamically. If the user is satisfied with the resource modification, Editres appends the modified resource string to a selected app-defaults file or to the user's .Xdefaults file.

Despite its beneficial features, Editres suffers from several disadvantages. For example, some applications require source code modifications or recompilation in order to work with Editres. In the dynamic editing mode, Editres fails if it cannot communicate with the application. Editres also tends to overcomplicate the editing process by presenting the user with an entire widget tree containing every widget used by an application. Often, the user is not interested in more than